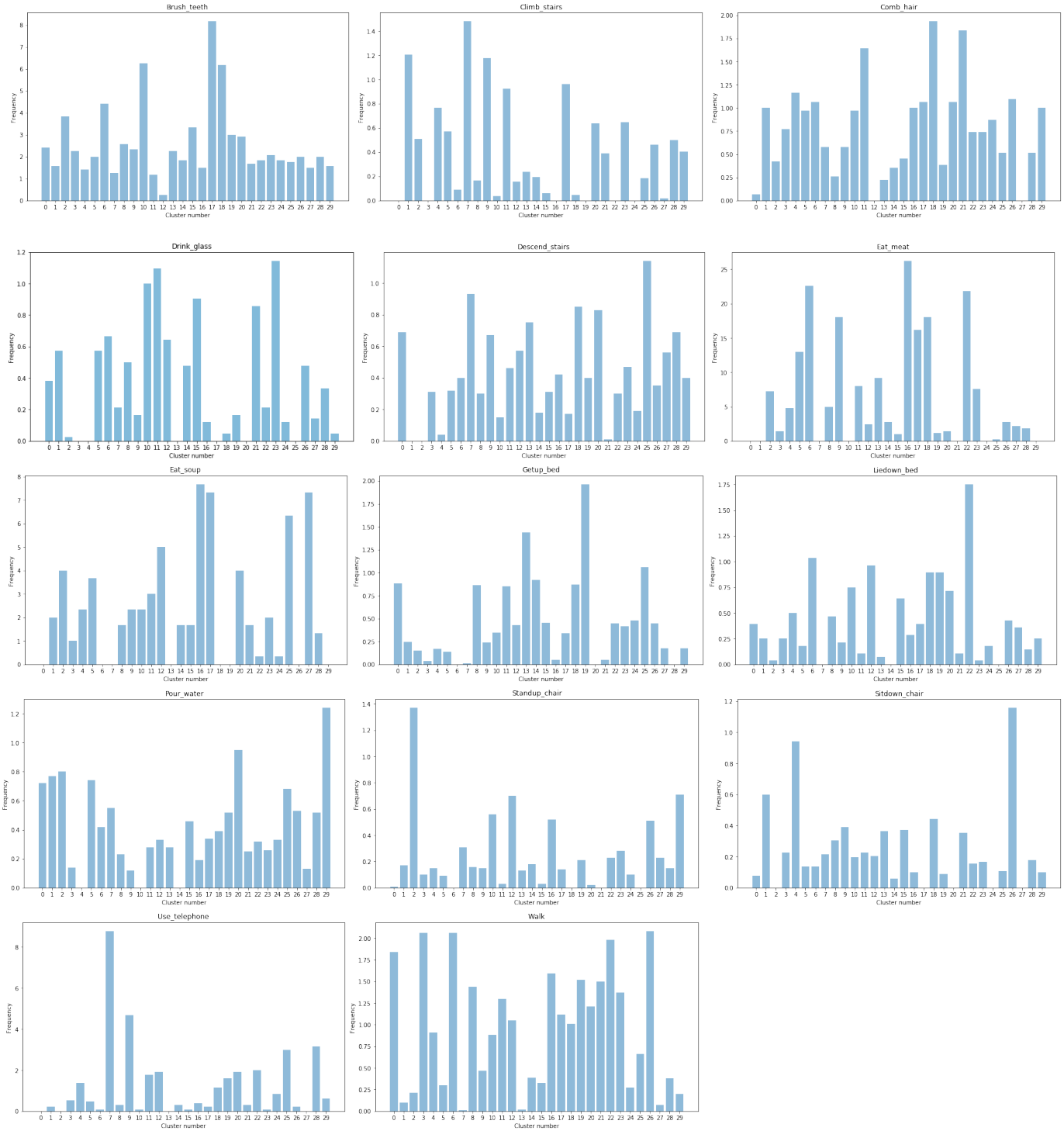
Page 1 Experiment table

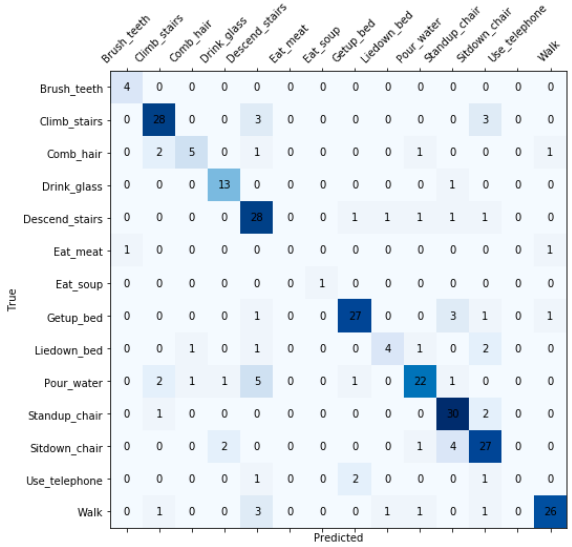
| Size of fixed length | Overlap (0-x%) | K-value | Classifier | Accuracy (avg) |
|---|---|---|---|---|
| 16 | 0 | 15 | random forest | 0.7665 |
| 32 | 0 | 30 | random forest | 0.7726 |
| 32 | 0 | 320 | random forest | 0.6534 |

Page 2 Histograms  k = 30, size of fixed length = 32

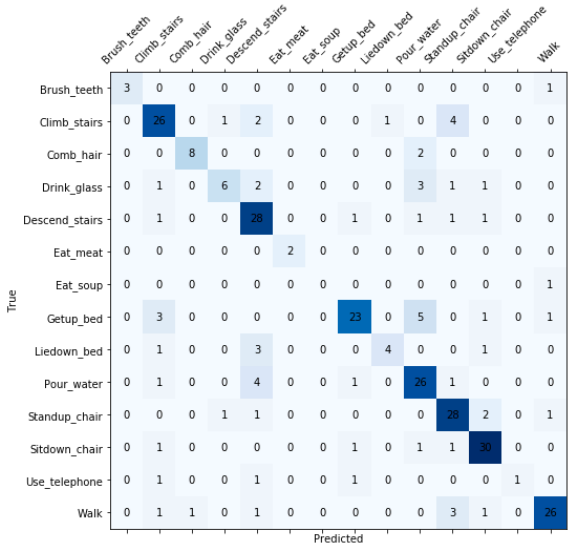


Page 3 Confusion matrix

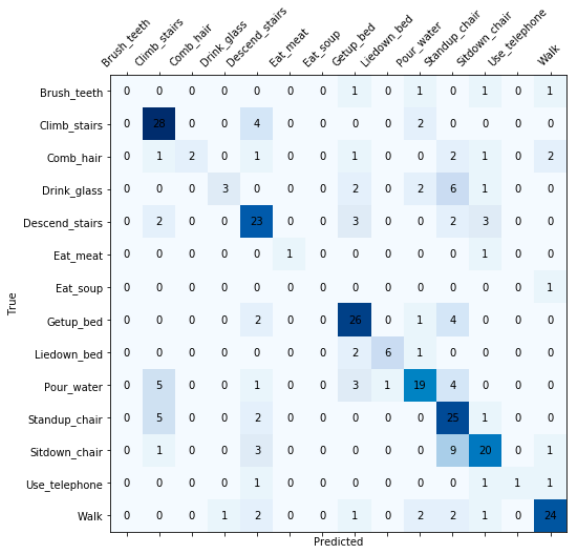
1. chunk size = 32, cluster size = 30




2. chunk size = 16, cluster size = 50




3. chunk size = 32, cluster size = 320



Page 4

## i) Segmentation of the vector

```
def get_chunk_vector(filelist, chunk_size):
    ##read all files in each folder and get segments of vector
    segment = []
    for filename in filelist:
        data = np.genfromtxt(filename, dtype = int)
        for idx, d in enumerate(data):
            if idx != 0 and idx % chunk_size == 0:
                tmp = np.array([])
                tmp = np.append(tmp, [data[idx - chunk_size: idx]])
                segment.append(tmp)
    segment = np.array(segment)
    return segment
```

## ii) K-means

```
import warnings
warnings.filterwarnings('ignore')
codebooks = []
##create idc with whole data each folder
def get_codebook(segment_list_act, cluster_size):##run kmeans on each segment vector for each folder
    for i in range(len(files)):
        center, label = kmeans2(segment_list_act[i], cluster_size)
        codebooks.append(center)
```

## iii) Generating the histogram

```
i = 0
j = 0
while j < 14 and i <= sum(files_length):
    hist_tmp = histograms[i : i + files_length[j]]
    i += files_length[j]
    y = np.arange(30)
    fig, ax = plt.subplots()
    fig.set_size_inches((10, 6))
    plt.bar(y, np.mean(hist_tmp, axis = 0), align = 'center', alpha = 0.5)
    plt.xticks(y, y)
    plt.ylabel('Frequency')
    plt.xlabel('Cluster number')
    plt.title(labels[j])
    plt.show()
    j += 1
```

## iv) Classification

```
from sklearn.metrics import confusion_matrix
accuracy = []
cm = []
for i in range(3):
    classifier = RandomForestClassifier(max_depth = 20, random_state = 0)
    X_train, X_test, y_train, y_test = train_test_split(histograms, histograms_labels,
                                                        test_size=0.33, stratify=histograms_labels)
    classifier.fit(X_train, y_train)
    predictions = classifier.predict(X_test)
    accuracy.append(accuracy_score(predictions, y_test))
    cm.append(confusion_matrix(y_test, predictions))
```

Page 5 All the source code

```
import glob
import matplotlib
import pandas as pd
import numpy as np
import os
import matplotlib.pylab as plt
from matplotlib import cm
from sklearn.ensemble import RandomForestClassifier
from scipy.cluster.vq import vq, kmeans, whiten, kmeans2
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
activities = {'Brush_teeth': 0,
              'Climb_stairs': 1,
              'Comb_hair': 2,
              'Descend_stairs': 3,
              'Drink_glass': 4,
              'Eat_meat': 5,
              'Eat_soup': 6,
              'Getup_bed': 7,
              'Liedown_bed': 8,
              'Pour_water': 9,
              'Sitdown_chair': 10,
              'Standup_chair': 11,
              'Use_telephone': 12,
              'Walk': 13}
```

```
labels = sorted(activities.keys(), key = lambda x: x[0])
```

```
files = [None] * len(activities)
train_files = [None] * len(activities)
test_files = [None] * len(activities)
path = '/Users/xinqu/Sandbox/CS498 Applied Machine Learning/HW/HW5/HMP_Dataset/'
def read_data(activities):
    for act in activities:
        f = glob.glob(path + act + '/*.txt')
        files[activities[act]] = f
read_data(activities)
```

```
files_length = []
for i in range(14):
    files_length.append(len(files[i]))
```

```
def get_chunk_vector(filelist, chunk_size):
    ##read all files in each folder and get segments of vector
    segment = []
    for filename in filelist:
        data = np.genfromtxt(filename, dtype = int)
        for idx, d in enumerate(data):
            if idx != 0 and idx % chunk_size == 0:
                tmp = np.array([])
                tmp = np.append(tmp, [data[idx - chunk_size: idx]])
                segment.append(tmp)
    segment = np.array(segment)
    return segment
```

```
segment_list_act = []##segments list by each folder
for i in range(len(files)):
    segment_list_act.append(get_chunk_vector(files[i], 32))
```

```
import warnings
warnings.filterwarnings('ignore')
codebooks = []
##create idc with whole data each folder
def get_codebook(segment_list_act, cluster_size):##run kmeans on each segment vector for each folder
    for i in range(len(files)):
        center, label = kmeans2(segment_list_act[i], cluster_size)
        codebooks.append(center)
get_codebook(segment_list_act, 30)
```

```
data_files = []
for act in activities:
    data_files.append([act])
```

```
histograms = np.array([])
histograms_labels = []
def get_histogram(files, codebooks, cluster_size, chunk_size):
    ###get histograms for each file in each folder
    global histograms, histograms_labels
    for i, j in enumerate(data_files):
        for act in j:
            for filename in files[i]:
                segment = [] ##get segment for each file
                data = np.genfromtxt(filename, dtype = int)
                for idx, d in enumerate(data):
                    if idx != 0 and idx % chunk_size == 0:
                        tmp = np.array([])
                        tmp = np.append(tmp, [data[idx - chunk_size: idx]])
                        segment.append(tmp)
                traincode, distortion = vq(segment, codebooks[i])
                hist = np.bincount(traincode, minlength = cluster_size)
                if len(histograms) == 0:
                    histograms = np.array([hist])
                else:
                    histograms = np.concatenate((histograms, np.array([hist])), axis = 0)
                histograms_labels.append(i)
get_histogram(files, codebooks, 30, 32)
```

```
from sklearn.metrics import confusion_matrix
accuracy = []
cm = []
for i in range(3):
    classifier = RandomForestClassifier(max_depth = 20, random_state = 0)
    X_train, X_test, y_train, y_test = train_test_split(histograms, histograms_labels,
                                                        test_size=0.33, stratify=histograms_labels)
    classifier.fit(X_train, y_train)
    predictions = classifier.predict(X_test)
    accuracy.append(accuracy_score(predictions, y_test))
    cm.append(confusion_matrix(y_test, predictions))
```

```
np.mean(accuracy)
```

```
0.7725631768953068
```

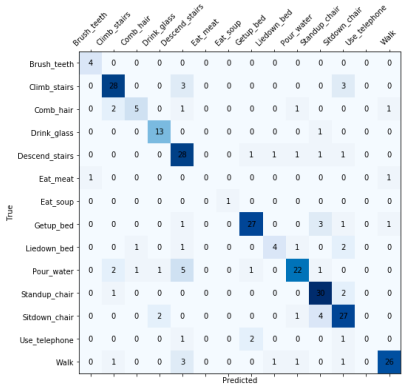
```
##avg error rate
1 - np.mean(accuracy)
```

```
0.22743682310469315
```

```
print(cm[2])
```

```
[[ 3  0  0  0  1  0  0  0  0  0  0  0  0  0]
 [ 0 31  0  0  0  0  0  0  0  0  1  2  0  0]
 [ 0  1  6  0  0  0  0  0  0  3  0  0  0  0]
 [ 0  0  0  6  0  0  0  3  0  0  1  2  0  2]
 [ 0  1  1  1 27  0  0  2  0  0  0  1  0  0]
 [ 0  0  0  0  0  0  0  0  0  1  0  0  0  1]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  1]
 [ 0  2  0  0  2  0  0 27  0  1  0  1  0  0]
 [ 0  1  2  0  2  0  0  0  3  1  0  0  0  0]
 [ 0  1  0  0  2  0  0  0  0 28  1  1  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 29  4  0  0]
 [ 0  1  0  0  1  0  0  2  0  0  0 30  0  0]
 [ 0  2  0  0  0  0  0  1  1  0  0  0  0  0]
 [ 0  2  0  0  1  0  0  3  0  0  1  0  0 26]]
```

```
fig, ax = plt.subplots()
fig.set_size_inches(8, 8)
ax.matshow(cm[0], cmap=plt.cm.Blues)
for i in xrange(14):
    for j in xrange(14):
        c = cm[0][j,i]
        ax.text(i, j, str(c), va='center', ha='center')
ax.set_xticks(np.arange(0, 14))
ax.set_xticklabels(labels, minor = False, rotation = 45)
ax.set_yticks(np.arange(0, 14))
ax.set_yticklabels(labels, minor = False)
plt.xlabel('Predicted')
plt.ylabel('True')
```



```
i = 0
j = 0
while j < 14 and i <= sum(files_length):
    hist_tmp = histograms[i : i + files_length[j]]
    i += files_length[j]
    y = np.arange(30)
    fig, ax = plt.subplots()
    fig.set_size_inches((10, 6))
    plt.bar(y, np.mean(hist_tmp, axis = 0), align = 'center', alpha = 0.5)
    plt.xticks(y, y)
    plt.ylabel('Frequency')
    plt.xlabel('Cluster number')
    plt.title(labels[j])
    plt.show()
    j += 1
```

References:


1. https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.vq.vq.html
   example code of vector quantization
2. https://docs.scipy.org/doc/scipy-0.16.0/reference/generated/scipy.cluster.vq.kmeans2.html built-in standard k-means library
3. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
   decision forest classifier in python
4. https://github.com/semerj/activity-prediction/blob/master/activity-prediction.ipynb
   example code to load data
5. https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html
   confusion matrix definition
6. https://stackoverflow.com/questions/40887753/display-matrix-values-and-colormap
   how to visualize the confusion matrix
7. Textbook